A major purpose of the Technical Information Center is to provide the broadest dissemination possible of information contained in DOE's Research and Development Reports to business, industry, the academic community, and federal, state and local governments.

Although a small portion of this report is not reproducible, it is being made available to expedite the availability of information on the research discussed herein.

1

LA-UR -89-3685

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

LA-UR--89-3685

DE90 003391

TITLE: **Creating the Next Generation Control System Software**

AUTHOR(S): **David E. Schultz**

SUBMITTED TO **International Conference Accelerator and Large Experimental Physics Control Systems Vancouver, British Columbia Canada October 30, November 3, 1989**

DISCLAIMER

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

Creating the Next Generation Control System Software

David E. Schultz
Los Alamos National Laboratory
Los Alamos, NM 87545

-Abstract-

A new 1980's style support package for future accelerator control systems is proposed. It provides a way to create accelerator applications software without traditional programming. Visual Interactive Applications (VIA) is designed to meet the needs of expanded accelerator complexes in a more cost effective way than past experience with procedural languages by using technology from the personal computer and artificial intelligence communities.

1 THE PROBLEM

The three main computer intensive phases of converting an idea for an accelerator into an accelerator producing beam for experimenters are: design, commission (calibrate), and operate (tune, monitor, trouble shoot). Unfortunately, in many cases, the computer support for these three phases is independent, i.e., no cross use of software, interfaces, or data is made. Creating the software for the control system for a new or expanded accelerator is a costly and difficult task. The size of such projects is often estimated in the 100's of man-years range. This software can be broken up into three basic levels.

1. Operating system interface
2. Library (utility support)
3. Applications programs

The operating system interface level contains the device drivers and routines that handle any communications protocol. At most installations, this level is kept as small as possible to avoid having to retrofit these features each time the manufacturer makes a major change to the underlying operating system. This level is the most sensitive to errors. A failure here often means that the entire system comes to a crashing halt.

The library or utility support level is the collection of utility routines to access data, give device commands, display output information, and obtain input data. This level may contain subroutines that convert device descriptors into data access paths to the device itself. It also contains a host of other programs designed to make the applications easier to write. Often the standards for getting and displaying data are done at this level.

The application program level is the largest. It contains all the man/machine interface programming and all the algorithms for manipulating the device data. It uses the library and operating system interface levels to obtain data, process it, and display the results. Applications programs are often written by a diverse group of people: operators, users, beam line physicists, professional programmers, and others.

Cross fertilization among design, calibration, tuning, operation, and diagnosis could speed the total software task, reduce cost of development and training, provide more powerful environment to solve the problems of getting an accelerator up, and reduce the time needed to produce physics results.

2 THE NEED FOR VIA

With the ever increasing cost of producing software and the increasingly complex accelerator collections needed to reach desired energies and intensities , software engineers must find ways to meet the needs to understand and control these complex accelerators more cost effectively.

Several advanced accelerators have been proposed. All require substantial software effort. The final design of any of these new machines will undoubtedly require an evolving sequence of accelerators, some built and commissioned separately, but all requiring coordinated timing and controls to operate successfully. Successful operation will not only include delivering the promised beam energies and intensities, but also limiting beam losses, which can result in unacceptable activation levels in a high intensity accelerator. The final design will also most likely allow for possible upgrades to the accelerators in both intensity and energy.

At a recent Advanced Hadron Facility (AHF) workshop [1], the effort required to produce the software needed to operate one large advanced accelerator facility was estimated at 240-280 man years. DeMarco has estimated [2] that the cost of delivering the first fully functional version of a software product is 39% of its total life-cycle cost. This means that the total life-cycle cost of the AHF software (based on 240 man years to develop) is about 600 man years. At current costs, the delivery of the initial fully functional system comes to about $43M and the total life-cycle cost in constant 1989 dollars is

over $100M. There are several factors that influence this cost:

1. the number of devices
2. the number of different devices
3. the number of accelerators
4. the complexity of the interactions between accelerators
5. the need for a consistent interface
6. the need to accommodate changes
7. the need for prompt maintenance (both bug fixes and changes)

Some of these problems can be addressed administratively. In particular, the number of different devices can be held to a minimum by developing standards for common devices and insuring that all devices of a given class adhere to the standard set. The number of devices, the number of accelerators, and the complexity of interactions between the accelerators are all physics questions that can not be altered by software engineers. Though it should be noted that attempts to cut costs of constructing an accelerator by reducing the number of diagnostic devices may be counter-productive. Any savings in construction costs may be quickly used up by increases in software, operations, and tuning. "What can't be measured, can't be controlled" [2] applies to controlling accelerators, as well as, controlling projects. The factors that are left that can be addressed by software engineers are the consistent interface, the design for change, and the prompt maintenance. One of the highest costs is the man machine interface. It has been estimated that 40-60% of a project goes into developing the user interface.

3 A PROPOSED SOLUTION

Some solutions have been suggested to reduce the estimated 240 man years of effort required to support the AHF [3]. Suggestions include windows, CASE, OOP, and Work Stations. Unfortunately, these suggestions address accidental not fundamental problems in software development [4]. For that reason, the expected gains will be modest. Rather than just patch the current software development approach and gain a small amount, it seems wise to take that small gain and also shoot for a quantum leap in leverage by fixing the right problem, i.e, allowing the sources of knowledge to implement standard maintainable programs without making those applications or operations experts become professional programmers.

The PC revolution of the late 1970's and early 1980's showed that many non-programmers could solve significant computer problems when given the right tool. VISICALC opened the door for a new way to specify an application to a computer without having to learn how to write traditional programs. Another recent product eliminates the need to learn to write Standard Query Language (SQL) formatted requests to obtain information from a data base management system (DBMS). The product is INTELLISCOPE from Intellicorp. It is a mouse and menu interface to select subsets of data from the data base and process it using a predefined set of mathematical and statistical tools. The Visual Interactive Applications (VIA) package adapts the same philosophical approach to provide a powerful tool for accelerator operators and designers. The adaptation uses the collection of devices to take the place of the data base and provides functions appropriate for accelerator operations instead of the mathematical or statistical functions.

This basic approach minimizes the amount of applications software needed to be written using traditional programming techniques by providing a high level environment tailored to creating solutions to accelerator control and operation problems. In this environment you don't write code; you assemble pieces (functions) needed and let the tool generate a program or interpret the functions. This approach provides leverage in solving the problem.

VIA reduces duplication of effort, provides a consistent interface, and supports the major computer intensive tasks of accelerator implementation. Using VIA to create applications has a four-fold benefit. First, more people can contribute to the creation of applications programs. Building an application is simple, straight-forward, and easy to learn. Second, there is a reduced need for communication between the application expert and the programmer because the application expert can be the creator of the "program". Third, all applications developed in the VIA environment meet the standards set up for input, output, error handling, device naming conventions, and visual presentation because VIA only generates applications that meet the standards. Fourth, all bookkeeping is done by the computer. Hence, there should be fewer errors and applications should be implemented more quickly. In addition, all of the applications created by VIA have the same look and feel regardless of who the "programmer" was.

3.1 Related Work

There has been a considerable amount of work done in user interface creation toolkits [5]. Several active research projects are exploring application frameworks [6,7] or even a visual programming environment [8]. All stop short of really providing applications without programming. This is because the functions provided in these packages are very low level to allow

for the solution of any general programming problem. They are intended for use by programmers to create traditional programs in an unrestricted way. VIA is highly tailored for the specific accelerator application. Neither a text editor nor a pay roll program can be written with VIA. It is, however, very well suited for solving accelerator control and operations problems because the high level functions available are tailored specifically to that task and to no other. By tailoring VIA to solve the accelerator control problem, the local computing power is used to help create the needed software.

3.2 Providing For Basic Needs

Man machine interfaces are critical for the success of an accelerator control system. Leveled direct manipulation is an approach that is natural, easy to learn, and is intuitive to use. It is based on a high resolution graphical/iconic representation of multiple levels of detail of multiple views of the various subsystems of the accelerator. A high resolution pointing device with multi-button and/or menu activated selection of various attributes and functions associated with selected device allows the operator to quickly and easily select what function he wants applied to which devices.

4 SOME FEATURES OF VIA

An applications development environment similar to VISICALC or INTELLISCOPE provides consistency in the user interface, standards for ease of maintenance, repair, and extension, and an environment where non-professional programmers can contribute productively to the creation of application software. Intellicorp's INTELLISCOPE product is a mouse and menu interface that allows selection of a subset of entries from a database and the subsequent processing of the collected entries. A similar approach is taken in VIA. VIA just substitutes the set of known devices on the accelerator for data base and provides functions appropriate to those devices.

4.1 Selection Features

VIA selection criteria:

1. by class
2. by sector
3. by function , e.g., vacuum, RF, cooling

4. by running mode

5. by energy

6. by injector

7. by ring

8. by beam pulse flavor and timing constraints

9. by devices out of tolerance

10. by device status, i.e., OK, broken, suspect, unknown

11. by present device reading (ranges)

Figure 1 shows a typical VIA interaction panel. Note the side board is a pallet of selection choices. Constructing an application with VIA can be thought of as similar to ordering from a Chinese menu. Select one from column A and one from column B.

4.2 Processing Features

The functions provided are related to the phase of the project.

4.2.1 Designing Needs

External to the control system beam line designers have long used computer support. Many different modeling programs are available (TRANSPORT ,TRACE, MARYLIE, MAD, etc) each has a different beam line input form, beam representation, capabilities (strengths and weaknesses), and output. A common interface to all modeling programs used in the design of a particular accelerator would be extremely useful. The task of exploring a large design space that is, maintaining the dependencies between different parts of the design, checking constraints, applying analysis (simulation), and cataloging alternate designs, is an immense, almost overwhelming, task.

VIA processing for design

1. transport lattice

2. cooling

3. acceleration

4. injection

5. extraction

6. floor layout

7. cost

8. schedule (PERT)

4.2.2 Commissioning Functions - During start-up of an accelerator beam line, the accelerator physicist tries to find errors in the system that cause the actual beam trajectory to deviate from the desired (design) trajectory. These errors are typically magnet calibration and construction alignment errors.

VIA processing for commissioning

1. find calibration errors

2. find mis-alignments of active elements

3. find offset errors of diagnostic devices

   adjust model to reality

5. cost

6. schedule (PERT)

4.2.3 Tuning Functions - Tuning of existing accelerators is often a long and arduous task.

VIA processing for tuning

1. sequence devices on

2. tune for a specific energy

3. tune for minimum loss

4. tune for maximum transmission

5. multiple injector tune

6. tune for major experiment

7. tune with constraints, e.g., reduce power consumption

4.2.4 Operating Functions - The operations staff needs to be able to tune the accelerator, change from one tune to another, maintain the status quo, i.e., adjust for power supply drift, and trouble shoot problems that arise.

VIA processing for operations:

1. monitor
2. record
3. check point (save current settings)
4. restore save set
5. interface (assign "knob")
6. document (for accounting purposes)
7. change tune (sequence to another accelerator state)
8. sequence devices off (shut down)

4.2.5 Diagnostic Functions - Expert systems have been used to diagnose accelerator problems[9].

VIA processing for diagnosis of problems

1. check for symptoms
2. identify failures
3. suggest remedies
4. log trouble report
5. call appropriate expert

4.3 Output Features

A wide variety of output formats are useful to users.

VIA output formatting options:

1. graphical - strip chart
2. graphical - X Y plot
3. tabular - e.g., DSP
4. historical - graphical or tabular
5. beam profiles
6. beam spot size/shape
7. archival - raw
8. archival - processed data

5 CONSIDERATIONS

5.1 Cost Considerations

One major concern on every ambitious software development project is cost. Though it is impossible to state precise figures, the OOP technique is known to be a cost effective method for software development and maintenance. A common operator interface should also reduce documentation and training costs. It is also expected that some of the support routines will be shared among the three main functional areas, thereby reducing the total cost for providing all the features. Maintenance cost should also be reduced because of the use of higher level tools and the reduced number of total modules need to support all the computer intensive accelerator applications. VIA should reduce development costs and simplify maintenance of applications. The use of higher level language for development of an accelerator control system has proven to be beneficial in the past. Explicitly defined knowledge is easier to change than implicit knowledge [10].

5.2 Size Considerations

Another major concern on software projects of this magnitude is the question- Is the project too big to successfully complete? There are numerous advantages to using an advanced software technology approach to the AHF software development project. The most important from the point of view of size is incremental development. Incremental development encourages prototyping with the goal of producing a working subset of the system as early as possible [11] This working subset:

1. provides a psychological boost to the developers
2. can be used to test parts of the system
3. gives users a way to provide feedback
4. allows for performance measurements.

In addition, this approach encourages reusable components, thereby, reducing the overall number of software objects.

6 SUMMARY

The versatility of VIA allows operators and beam line physicists, as well as programmers, to create useful functionality. When a particular mix of selection, processing, and output format is found to be generally useful, it can be captured for general use by saving the steps used. This allows incremental building of a set of functions that can be used with a minimum of typing and/or mouse actions by the operator and a minimum number of constraints on the builders. It does make the Control System programmers' job more difficult because they must develop general tools not specific applications. However, the overall effect is to provide a more powerful, more consistent system that supports more different applications (from more minds) at a lower overall cost than traditional approaches. The resulting collection of applications is more consistent because they are built with the same tool. Consistency of the operator interface and the definition of applications programs is provided at the tool level and not by way of administrative controls. If a new technology provides a better interface or more effective processing, the tool can be upgraded to take advantage of it and provide the new technology to all the "applications" automatically.

References

1] AHF 1989 Workshop, Los Alamos, NM , Feb 20-24, 1989

2] Controlling Software Production, Tom DeMarco, Yourdon Press, New York, 1982

3] "An Advanced Hadron Facility: A Combined Kaon Factory and Cold-Neutron Source," H. Thiessen, Proceedings of the 1987 Particle Accelerator Conference, Washington, D.C., March 1987.

4] "No Silver Bullet, Essence and Accidents of Software Engineering," Frederick Brooks, COMPUTER, April 1987, p10-19

5] "A User Interface Toolkit Based on Graphical Objects and Constraints," Pedro A Szekely and Brad A. Meyers, OOPSLA'88 Proceedings, p 36-45

6] "ET++ -An Object-Oriented Application Framework in C++," Andre Weinand, Erich Gamma, Rudolf Marty, OOPSLA'88 Proceedings, p 46-57

7] "Transportable Applications Environment (TAE) PLUS," Martha Szczur, Philip Miller, OOPSLA'88 Proceedings, p 58-70

8] "Fabrik A Visual Programming Environment," Dan Ingalls, Scott Wallace, Yu-Ying Chow, Frank Ludolph, Ken Doyle, OOPSLA'88 Proceedings, p 176-190

9] "A Fault Diagnosis Expert System for CERN using KEE".Skarek, P., Malandain, E., Pasinelli, S., Alarcon, I.,, CERN/PS 88-12 (CO)

10] "Expert Systems: Perils and Promise",Bobrow, D., Mittal, S., Stefik, M.,

CACM, September 1986 , Vol 29, No 9.. pp 880-894

11] "Software Development of Real-Time Systems",Gomaa, H. , CACM, July 1986 , Vol 29, No 7., pp 657-668